DOCUMENTAZIONE CASO DI STUDIO

METODI AVANZATI DI PROGRAMMAZIONE

A.A.2023-2024

NOME PROGETTO:

“THE HOUSE OF THE RIDDLE”

Componenti:

Maldera Antonella [758380]

Miticocchio Maria Sara [758399]

Termine Christian [774864]

Sommario

1. L’avventura testuale

1.1 Spunti e trama del gioco

Per il nostro caso di studio abbiamo deciso di progettare un’avventura mista, sia testuale che grafica.

Ti risvegli in una casa abbandonata. Le finestre sono coperte di polvere e le porte cigolano a ogni movimento. Non ricordi come sei arrivato qui, ma capisci ben presto che devi trovare un modo per uscirne. Per riuscirci, dovrai esplorare la casa, raccogliere oggetti, scoprire segreti e risolvere enigmi per svelare la storia di questo luogo misterioso.

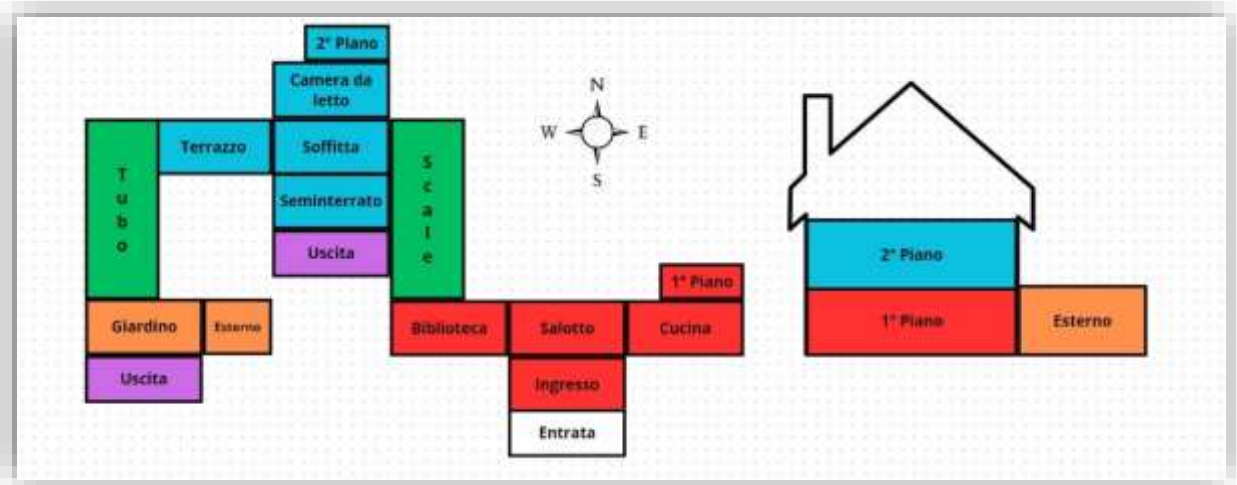
Eccoti, dunque, all’inizio della tua avventura: ti ritroverai catapultato all’ingresso di una casa abbandonata.
ALLERTA SPOILER: È FORTEMENTE SCONSIGLIATO PER CHI NON HA ANCORA COMPLETATO IL GIOCO LEGGERE DA QUI IN POI, IN QUANTO VERRA’ SPOILERATA LA SOLUZIONE DELL’AVVENTURA.
Obiettivo e risoluzione
Il tuo obiettivo è uscire vivo dalla casa dell'enigmista e tornare alla realtà. Per farlo, dovrai esplorare la casa e le sue stanze. Una volta arrivato in soffitta, l'enigmista ti offrirà due vie di uscita.
Per la prima uscita, dovrai dirigerti a Ovest verso il terrazzo. Da lì, potrai scendere nel giardino utilizzando una tubatura. Una volta in giardino, troverai un cancello che potrai aprire inserendo un codice. Questo codice, 1234, lo

avrai scoperto illuminando un libro in biblioteca con una torcia trovata in cucina.
Per la seconda uscita, dovrai aprire una porta chiusa utilizzando un piede di porco trovato nel baule della camera da letto. Questo ti porterà nel seminterrato, dove dovrai rispondere correttamente ad almeno due delle domande che ti verranno poste per uscire sano e salvo.

Pericoli e Morte

Durante il percorso, ci sono quattro modi per morire.
Il primo modo è in salotto: se usi una candela e dei fiammiferi per illuminare la stanza, vedrai un pulsante. Premendolo, verrai trafitto da frecce velenose.
Il secondo modo è in cucina: se usi un coltello per aprire un cassetto, questo esploderà.
Il terzo modo è in giardino: se inserisci il codice errato più di tre volte al cancello.
Il quarto modo è nel seminterrato: se non rispondi correttamente ad almeno due domande.

1.2 Mappa



1.3 Come si gioca

Al giocatore è permesso il movimento e il passaggio da camera a camera con i quattro punti cardinali:
nord,sud,est e ovest: per il passaggio da piano a piano (1°piano→2°piano, Terrazzo→Giardino) sono usati degli strumenti che sono, rispettivamente, scale e tubo.
Per ogni stanza, il giocatore avrà automaticamente una descrizione della stanza, potrà usare il comando "guarda" per essere informato sull'eventuale presenza di oggetti nella stanza ed ottenere indicazioni o suggerimenti da usare successivamente nel gioco.

1.4 Lista dei comandi a disposizione

All'interno della nostra avventura testuale sono presenti due categorie di comandi: comandi di movimento e comandi d'azione.
I comandi di movimento sono i seguenti:

nord	N-Nord-NORD-nord
sud	S-Sud-SUD-sud
est	E-Est-EST-est
ovest	O-Ovest-ovest

Per i comandi d'azione invece:

APRI	apri
PRENDI	prendi-afferra-raccogli
GUARDA	guarda-vedi-trova-cerca-descrivi-osserva
USA	utilizza-combina
PREMI	inserisci-schiaccia-attiva-spingi
INVENTARIO	inv-inventario

END	end-fine-esci-muori-exit
INSERISCI	inserisci
HELP	help

APRI: permette l'apertura di oggetti (tenda, baule, armadio).
PRENDI: permette di raccogliere gli oggetti.
GUARDA: fornisce una descrizione della stanza e indicazioni/indizi sugli oggetti prendibili presenti all'interno della stanza.
USA: combina più oggetti e ne permette l'utilizzo
PREMI: permette di premere oggetti (pulsante).
INVENTARIO: mostra la lista con gli oggetti raccolti fino a quel punto.
END:permette di uscire dal gioco in qualsiasi momento.
INSERISCI: permette di inserire il codice per aprire il cancello e vincere il gioco
HELP: una volta chiamato, mostra tutti i possibili comandi.

(nella colonna di sinistra delle tabelle riportate sopra abbiamo il nome completo del comando, in quella di destra tutti gli alias con cui puo essere richiamato. Inoltre abbiamo reso i comandi case-sensitive.)

2. Aspetti tecnici

2.1 Architettura del sistema

L'avventura, scritta interamente in Java, presenta al proprio interno un'architettura tale da permetterci di lavorare in modo organizzato e idoneo al raggiungimento degli obiettivi prefissati.

La classe principale per il funzionamento dell'avventura è definita Engine che richiama la classe GameDescription. La classe GameDescription è una classe astratta in quanto prevede i metodi astratti init(), nextMove () e getWelcomeMsg (): questi metodi hanno la finalità di poter essere riutilizzabili permettendo a chi creerà un'avventura di poterli reimplementare basandosi sul GameDescription.

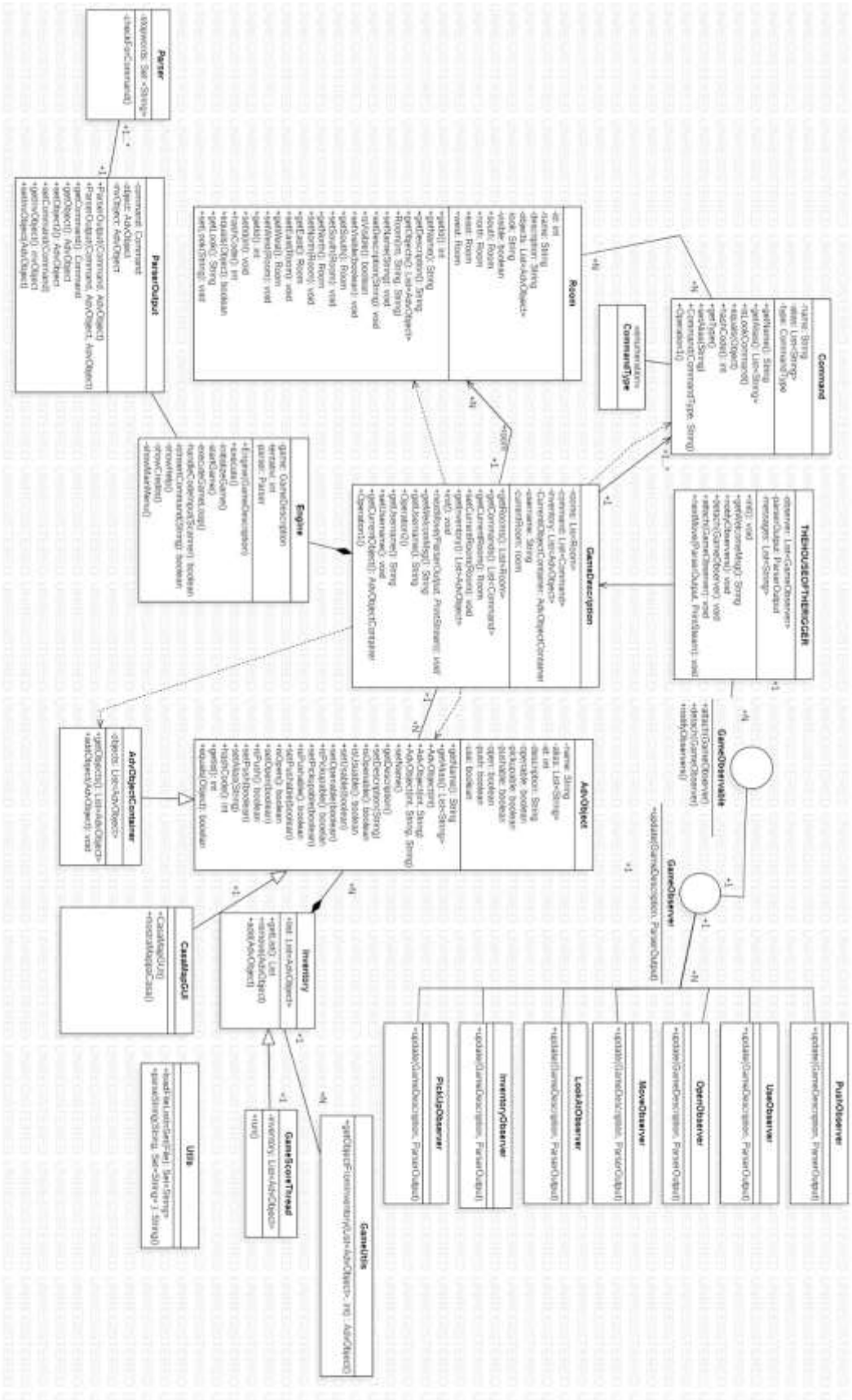
L'interfaccia GameObservable è responsabile dell'istanziamento, cancellazione e aggiornamento degli Observer, ovvero ogni oggetto osservabile nel contesto del gioco. L'interfaccia GameObserver notifica i vari Observer di nuove informazioni.

La classe GameScoreThread estende la classe Thread, una classe della libreria Java, ed è responsabile del calcolo e della comunicazione del punteggio del gioco nel caso si volesse uscire volontariamente, sovrascrivendo il metodo run() ereditato dalla classe Thread.

La classe GameUtils restituisce un oggetto dall'inventario basato sul suo ID. La classe Utils divide una stringa in token rimuovendo eventuali stopwords contenute all'interno del file Stopword, mediante metodo parserString.

La classe CasaMapGUI estende la classe JFrame che si occupa principalmente della rappresentazione grafica della mappa della casa e della sua stampa.

2.2 Diagramma delle classi



2.3 Specifica

LISTA

Per quanto riguarda la specifica algebrica, dato l'utilizzo di tre liste (stanze, oggetti e comandi), abbiamo deciso di prendere come oggetto la lista.
Una lista è una struttura dati algebrica lineare che rappresenta una sequenza ordinata di elementi, ognuno dei quali ha una posizione univoca.
Formalmente la lista comprende $a_1, a_2, a_3, \dots, a_n$ dove:

- Ogni elemento ha un indice o posizione
- Gli elementi possono essere ripetuti
- L'ordine degli elementi è significativo.

Le liste possono essere di 3 tipi:

1. Lista Singolarmente Collegata (Singly Linked List): Ogni elemento punta al successivo.
2. Lista Doppiamente Collegata (Doubly Linked List): Ogni elemento punta sia al successivo che al precedente.
3. Lista Circolare (Circular Linked List): L'ultimo elemento punta al primo, formando un ciclo.

Le operazioni principali relative alla lista sono: CreaLista per la creazione della lista vuota, ListaVuota per controllare se la lista contiene qualche elemento, LeggiLista per la lettura sequenziale della lista, FuoriLista che aggiorna la lista estraendo il primo elemento, InLista che aggiorna la lista inserendo l'elemento all'ultimo posto.

SPECIFICA SINTATTICA

Tipi: lista, elem, booleano, int

Operatori

CreaLista ()→lista
ListaVuota (lista)→booleano
LeggiLista (lista)→elem
FuoriLista (lista)→lista
InLista(lista, elem)→lista
IndiceElem (lista,elem)→int
isElem (lista,elem)→booleano

SPECIFICA SEMANTICA

Tipi:

lista= insieme delle sequenze di elementi di tipo elem
booleano= insieme dei valori di verità {vero,falso}
int= insieme dei valori di tipo intero

Operatori

- CreaLista()= L'
  Post: L' sequenza vuota
- ListaVuota (L)=b
  Post: b=vero se L=Λ, b=falso altrimenti
- LeggiLista (L)=a
  Pre: $L = a_{1,}a_{2,}a_3, \dots, a_n \; e \; n \geq 1$
  Post: $a = a_1$
- FuoriLista (L)=L'
  Pre: $L = a_{1,}a_{2,}a_3, \dots, a_n \; e \; n \geq 1$

Post: $L' = a_{2,}a_3, \dots, a_n \; se \; n > 1$,

L' =Λ se n=1

- InLista (L, a)= L'
  Post: L'=a se n=0, $L = a, a_{1,}a_{2,}a_3, \dots, a_n, \; se \; n > 0$ altrimenti
- isElem (L,a)= b
  Pre: $L = a_{1,}a_{2,}a_3, \dots, a_n \; e \; n \geq 1$
  Post: b= vero se $a \in L$; b=falso altrimenti
- IndiceElem (L,a)= i
  Pre: $L = a_{1,}a_{2,}a_3, \dots, a_n \; e \; n \geq 1$

  Post: i=-1 se isElem(L,a)= falso, i=n altrimenti

COSTRUTTORI ED OSSERVATORI

Osservatori	Costruttori	
	CreaLista()	InLista(lista, elem)
ListaVuota (lista')	True	false
LeggiLista(lista')	Error	lista
FuoriLista (lista')	error	lista
isElem(lista',elem')	error	True **se** elem'==elem
IndiceElem(lista',elem')	error	Int **se** elem'==elem

2.4 Dettagli di implementazione

Parser

Ogni avventura con ottimo gameplay è supportata da un parser.

Abbiamo previsto un package Parser che conta due classi, “Parser.java” e “ParserOutput.java”, che combinate permettono di analizzare le stringhe testuali. Nello specifico, queste due classi tokenizzano ogni parola su ogni spazio bianco e crea una coda di token.

Una volta tokenizzata la stringa in input, se uno dei token corrisponde a un comando o ad un oggetto, esso assegna un valore ad una variabile specifica: in caso contrario, viene considerata una "stop word" e quindi automaticamente scartata.

Nell’atto pratico, il giocatore potrà quindi dare come input “vai ad est”: le parole “vai” e “ad” saranno eliminate in quanto stopwords e la parola “est” sarà riconosciuta come comando. Lo stesso ragionamento verrà applicato per gli oggetti: se il giocatore darà come input “apri quel baule” saranno riconosciute solo le parole “apri” come comando e “baule” come oggetto contenitore.

Observer

Per gestire l’implementazione degli osservatori abbiamo deciso di creare un package chiamato “Impl” che prevede una classe per ogni comando. Ogni classe definita come “comando+Observer.java” si occupa dell’aggiornamento dell’Observer e stampa di messaggi riguardo quest’ultimo,

preceduto da una serie di controlli specifici per ogni classe.
Ad esempio, la classe “PickUpObserver.java” stampa un diverso output in base al risultato di alcune condizioni:

- Inserimento effettivo del comando “PICK_UP”
- Se l’oggetto specificato effettivamente esiste
- Se l’oggetto è prendibile

In caso di effettivo raccoglimento dell’oggetto, successivamente viene aggiunto nell’inventario e rimosso dalla stanza in questione. Come programmazione difensiva, sono previsti anche messaggi nel momento in cui l’inventario contiene tutti gli oggetti prendibili e nel momento di raccoglimento di uno dei due (o più) oggetti prendibili.

Nel package è presente una classe di nome “THEHOUSEOFTHERIGGER.java” ci permette di creare tutte le stanze, tutti gli oggetti e i rispettivi movimenti, elaborando risposta anche in caso di movimento non possibile, le descrizioni di ogni stanza (visibili una volta usato il comando “guarda”), il posizionamento di tutti gli oggetti nelle varie room. Abbiamo, inoltre, previsto e tenuto conto di tutti gli alias che il giocatore potrebbe utilizzare.

Mappa

Un’altra classe presente all’interno del package “Impl”, di nome “CasaMapGUI.java”, che estende la classe JFrame,

si occupa della creazione e stampa della mappa raccolta durante il gioco.